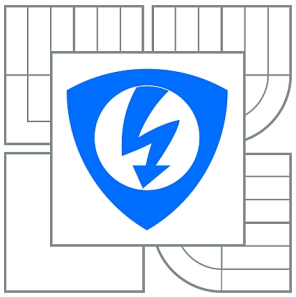
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE PROTOKOLU SIP

SIP PROTOCOL IMPLEMENTATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN DUŠEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR ČÍKA, Ph.D.

SUPERVISOR

BRNO 2011

Zadání

Abstrakt

Tato diplomová práce se podrobně zabývá protokolem SIP – způsobem komunikace dvou entit, různými typy přenášených zpráv a jejich obsahem. Také je představeno několik knihoven SIP, z nichž některé byly použity pro vývoj aplikace pro audio/video konference. Práce popisuje kompilaci vybraných knihoven OPAL a PTlib pro Windows 7 Professional (64-bit) a popisuje problémy vyplývající z neúplných informací autorů knihoven a prezentuje upravený návod. Na závěr se práce zaměřuje na konkrétní cesty vývoje zmiňované aplikace.

Klíčová slova

SIP, SDP, RTP, RTCP, klient, server, audio, video, videokonference, registrar server, proxy server, C++, aplikace, OPAL, PTlib, návod, vývoj, pojmenované roury

Abstract

This Master's thesis deals in detail with the SIP protocol – a method of communication between two entities, various types of transmitted messages and their content. Few SIP libraries are introduced and two of them are used for development of an application for audio/video conference-calls. Compilation of OPAL and PTlib libraries for Windows 7 Professional (64bit) is described, and problems resulting from lack of information provided by authors. New improved "how to build" is presented. At the end, paper focuses on several ways of development of mentioned application.

Key words

SIP, SDP, RTP, RTCP, client, server, audio, video, video conference, registrar server, proxy server, C++, application, OPAL, PTlib, how to, manual, development, named pipes

Bibliografická citace práce

DUŠEK, M. *Implementace protokolu SIP.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 46 s. Vedoucí diplomové práce Ing. Petr Číka, Ph.D..

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Implementace protokolu SIP“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení 152 trestního zákona č. 140/1961 Sb.

V Brně dne 22. 5. 2011

_____

Martin Dušek

Poděkování

Děkuji vedoucímu diplomové práce Ing. Petrovi Číkovi, Ph.D. a Ing. Jiřímu Přinosilovi, Ph.D. za velmi věcnou metodickou pomoc, cenné rady při zpracování této práce a její odbornou konzultaci.

_____

Martin Dušek

Obsah

Seznam obrázků

Úvod

SIP (*Session Initiation Protocol*) patří mezi větší množství protokolů používaných pro multimediální komunikaci v IP síti, ať pro samotné audio, video, či jejich kombinaci. Na rozdíl od jiných se však nezabývá přímo přenosem definovaných dat, ale pouze navázáním spojení a výměnou informací mezi oběma stranami, dohodnutím použitých verzí protokolů a dalších parametrů přenosu. SIP byl navržen jako protokol aplikační vrstvy ISO/OSI modelu a je nezávislý na spodních vrstvách. Data tak mohou být přenášena pomocí spojované služby TCP (*Transmission Control Protocol*), nespojované UDP (*User Datagram Protocol*), či služby SCTP (*Stream Control Transmission Protocol*). Jedná se o textově orientovaný protokol a využívá prvků HTTP (*HyperText Transfer Protocol*) a SMTP (*Simple Mail Transfer Protocol*). Sdílí také jejich jednoduchost a přehlednost. Samotná komunikace probíhá dle modelu server–klient a v síti se kromě těchto dvou prvků může nacházet také proxy, registrar a redirect server.

Veškeré informace jsou mezi jednotlivými entitami přenášeny pomocí několika druhů zpráv, které budou detailně popsány v teoretické části této práce, stejně tak jako detaily komunikace dvou koncových uzlů i ostatních možných konfigurací v síti. Druhá část se zaměří na praktickou stránku protokolu SIP, možnosti jeho implementace do aplikace za použití některých knihoven. Aplikace by měla být zaměřená na audio a video hovory, včetně konferenčních. Vývoj však přináší mnohé komplikace a i těm bude věnována část práce a může tak dobře posloužit i ostatním vývojářům.

1 Základy SIP

Pro pochopení SIP je vhodné alespoň nahlédnout do jeho historie, však bezpodmínečně nutné je prozkoumat jeho elementární prvky, formát a definice. Bez těchto mnohdy nestravitelných znalostí lze protokol jen těžko implementovat. Nejlepším způsobem pro poskládání mozaiky získaných informací a pro plné pochopení SIP je uvedení názorných ukázek. Proto budou uvedeny situace, které se mohou vyskytnout při nasazení tohoto protokolu a to v různých konfiguracích definovaných prvků. Na příkladech práce detailně popíše způsoby komunikace v těchto variantách a bude se zabývat jednotlivými typy přenášených zpráv, jejich obsahem a významem vypsaných parametrů.

1.1 Stručná historie SIP

K vývoji tohoto protokolu vedly snahy zjednodušit VoIP (*Voice over Internet Protocol*) signalizaci. Jak uvádí [2], verze 1.0 byla vydána a popsána v roce 1997 standardizační skupinou IETF (*Internet Engineering Task Force*) jako tzv. Internet–Draft (*ID*) – tedy jako koncept. V roce následujícím byl jako ID publikován SIP 2.0, který přinášel mnoho významných změn a oprav. O další rok později byl vydán RFC dokument číslo 2543, který byl následně v roce 2000 doplněn o několik záplat a dalších specifikací a označován jako RFC 2543 „bis“. V roce 2002 pak vyšly i další dokumenty, které SIP doplňovaly o další funkcionality.

V tuto chvíli je poslední aktuální dokumentací stále RFC 3261 ([1]) a SIP 2.0 se také stal oficiální součástí multimediální podskupiny IP protokolu.

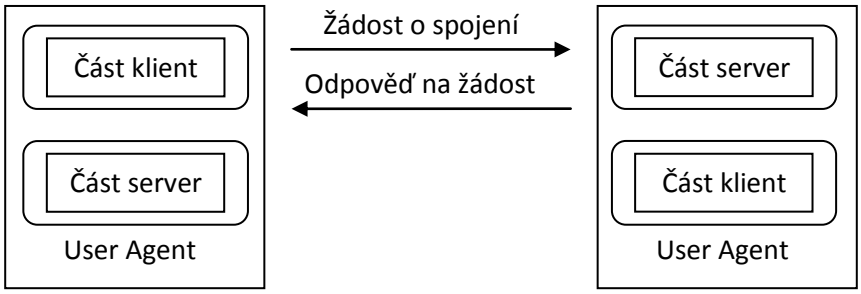
1.2 Struktura SIP

SIP definuje čtyři prvky, které spolu v síti mohou komunikovat. Jedná se o tyto entity:

- user agent
- proxy server
- redirect server
- registrar server

Každý z těchto prvků plní svou specifickou funkci v celém systému. Všechny mohou být rozprostřeny v síti nebo mohou být sloučeny na jediném serveru. Následující obrázky jsou ilustrativní a bez číselných kódů zpráv, které budou uvedeny a popsány později.

User Agent (*UA*) je koncové zařízení, které kombinuje obě funkce zmíněného modelu klient–server. Klientská část UA musí být schopna inicializovat audio/video/text spojení typu bod–bod a jeho serverová část odpovědět na tyto výzvy od jiného UA, jak je ukázáno na Obr. 1. Jeho dalším úkolem je pak registrace k registrar serveru pro účely mnohabodového spojení (konferencí).

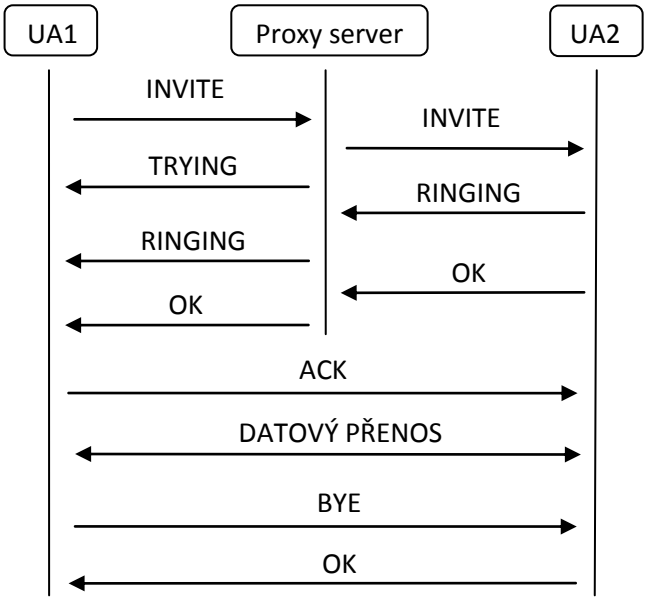


Obr. 1 – Znázornění úloh klientské a serverové části UA

Proxy server je bezpečnostní prvek na okraji sítě a ve vztahu k SIP představuje prostředníka mezi UA a zbytkem sítě. Přijímá žádosti od UA a přeposílá je dalším prvkům v síti (ať už dalším serverům nebo přímo hledanému UA). Když UA1 zašle zahajovací zprávu k UA2, dorazí k proxy serveru, který zjistí kde se UA2 nachází, zprávu přepošle a UA1 odešle odpověď, že se pokouší s UA2 spojit. Pokud zpráva k UA2 dorazí v pořádku, je odeslána odpověď zpět proxy serveru (ať již kladná nebo záporná), který ji opět přeposílá UA1. Další zprávy pak mohou probíhat přímo mezi UA, včetně datového přenosu, jak je naznačeno na Obr. 2. Významy jednotlivých zpráv budou vysvětleny později. Proxy server může být dvojího typu.

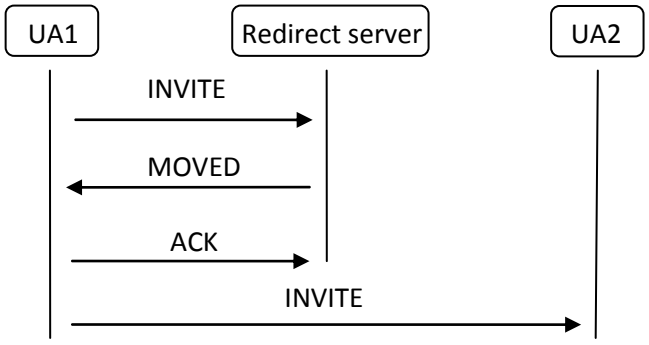
Statefull zaznamenává všechny zprávy a celá komunikace dvou entit v síti prochází přes něj. Je vhodný například pro monitorování či řízení.

Stateless proxy obsluhuje pouze úvodní přenosy zpráv a jakmile se oba UA zkontaktují, komunikace může probíhat přímo, bez proxy jako prostředníka. Podle [3] může proxy server zajišťovat také autentizaci (ověření uživatele), autorizaci (zda má uživatel právo provádět hovory) a aplikaci bezpečnostní politiky.



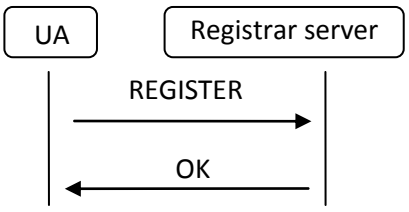
Obr. 2 – Komunikace přes proxy server

Redirect server plní jednoduchou funkci. Na žádosti o zahájení spojení s UA2 odpovídá alternativní adresou (cestou) vedoucí k hledané entitě, která se již na původním místě nenachází. Schéma takovéto komunikace je na Obr. 3.



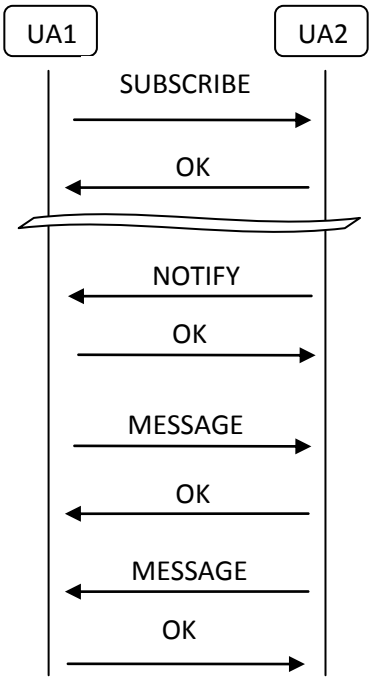
Obr. 3 – Použití redirect serveru

Registrar server slouží k registraci účastníků konferenčních hovorů. Komunikace je velmi jednoduchá – jak ukazuje Obr. 4.



Obr. 4 – Registrace UA

SIP lze také použít pro tzv. **Instant Messaging** (*IM*), tedy pro zasílání zpráv v téměř reálném čase. Součástí je také zjišťování prezence entity přihlášením se k odběru (pomocí zprávy SUBSCRIBE) změn stavů druhé entity, jak naznačuje Obr. 5. Zpráva NOTIFY oznamuje změnu stavu UA2 (například OFFLINE / ONLINE), zprávy MESSAGE pak přenášejí přímo krátké textové zprávy. Není tedy třeba navazovat spojení na přenos takto malých objemů dat.



Obr. 5 – Instant messaging v SIP

2 SIP zprávy

Pro potřeby komunikace jsou v SIP definovány dva druhy zpráv – žádosti a odpovědi (requests a responses). Jak již bylo řečeno, žádosti vysílá klient a odpovědi server. Obě tyto části jsou pak implementovány u každého prvku, který byl definován v předchozí podkapitole. Následující podkapitoly budou věnovány zvlášť žádostem a zvlášť odpovědím. U obou bude popsána struktura takové zprávy, uvedeny příklady podle [2] a vzhledem k jednoduchosti a přehlednosti bude podrobně vysvětůen i obsah několika z nich. Je také důležité předem zmínit význam dialogů a transakcí. Transakce je výměna zpráv mezi dvěma UA, které se týkají jednoho úkonu. Například transakce zahájení spojení bude zahrnovat zprávy od INVITE až po zprávu OK. Další transakce bude zahrnovat zprávy k ukončení spojení BYE – OK. Dialog se pak skládá z několika transakcí, od zahájení přes změnu až po ukončení spojení.

2.1 Žádosti

Zprávy obsahují metodu, významné pole, kterým začínají. Metody jsou vždy psány velkými písmeny a v [1] jich je popsáno celkem šest. Několika dalším se věnují ostatní RFC, zmíněné na konci kapitoly. Žádost se skládá z metody, hlavičkových polí a jejich hodnot. Některé mohou obsahovat i tělo zprávy s dalším textem.

2.1.1 INVITE

Zpráva pro inicializaci komunikace, specifikaci parametrů přenosu mezi dvěma UA. Celá zpráva vypadá například takto:

```
INVITE sip:marconi@radio.org SIP/2.0
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
Max-Forwards: 70
To: G. Marconi <sip:Marconi@radio.org>
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: 123456789@lab.high-voltage.org
CSeq: 1 INVITE
Subject: About That Power Outage...
Contact: <sip:n.tesla@lab.high-voltage.org>
Content-Type: application/sdp
Content-Length: 158
```

```
v=0
o=Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Za hlavičkou INVITE následuje adresa UA, kterému bude zpráva zaslána, a na konci je uvedena verze použitého protokolu (v tomto případě SIP 2.0).

Následuje pole **Via**, obsahující opět verzi SIP a protokol, jenž má být použit pro přenos zprávy. Dále je uvedena adresa a port samotné odesílající entity a na závěr je identifikátor *branch* – ten obsahuje unikátní kód, který jednoznačně určuje transakci, do které zpráva patří. Každá entita přeposílající zprávu sítí přidává vlastní pole Via, aby bylo možné postupně přeposílat odpověď zpět. Toho se využívá i pro detekci smyček.

Pole **Max–Forwards** slouží k omezení počtu směrování například při vzniku smyčky. Každý prvek, který zprávu přeposílá, dekrementuje hodnotu v tomto poli. Pokud je hodnota nulová, odešle zpět chybové hlášení. Pole může nabývat hodnot 0–255.

Další pole **To** a **From** určují příjemce a odesílatele zprávy pomocí URI (jednoznačný identifikátor UA uvedený mezi < >) a spolu s polem **CALL–ID** jednoznačně popisují dialog. Pole From navíc může nést i zobrazované jméno (uvedené v uvozovkách). První dvě zmíněná pole navíc obsahují tzv. *tag*, který je náhodně generován příslušnou stranou a jednoznačně určuje jeden hovor. Při posílání první INVITE zprávy je tag obsažen pouze u pole From, u pole To prozatím chybí a je vygenerován protistranou a doplněn až při odpovědi.

Doposud zmíněná hlavičková pole představují minimální obsah zprávy INVITE nutný pro sestavení hovorového spojení. Mohou být označena jako pole povinná, ostatní pak představují pole volitelná. Ta nesou doplňující informace nebo jiné požadované parametry, například při použití v jiných aplikacích než jsou hovory.

Subject má stejný význam jako u emailové zprávy, tedy uvádí důvod žádosti o hovor. **Contact** je pole obsahující zpáteční adresu UA, kterou lze použít pro přímé odpovědi, pokud by zpráva INVITE procházela přes proxy servery. **CSeq** uvádí o jaký typ zprávy (metodu) se jedná a jeho hodnota je shodná pro žádosti a odpovědi z jedné transakce.

Pole **Content–Type** určuje typ aplikace, která je obsažena v SDP zprávě a **Content–Length** udává počet oktetů v této zprávě. Pokud zpráva tuto část neobsahuje, nemusí být první pole ve zprávě obsažené, ale druhé musí mít hodnotu 0. V uvedeném příkladu je SDP část uvedena a tedy i její obsah bude popsán. Skládá se, podobně jako hlavní část INVITE, z proměnných (hlaviček) a přiřazených hodnot. Rozdíl pak spočívá v zápisu – proměnné jsou uvedeny s malým písmenem. Jejich významy jsou vypsány v Tab. 1, přeložené z [2]. Za každou hodnotou jsou pak dva oktety, které označují konec řádku. Tato část zprávy je důležitá z toho důvodu, že SIP sám o sobě nijak nedefinuje typ přenášeného média, kterým může být jak audio, video či data her a jiných zvláštních aplikací.

Tab. 1 – Význam parametrů SDP části zprávy INVITE

symbol	význam	povinnost /volitelnost
V	verze protokolu	P
O	identifikátor volajícího	P
S	název relace	P
C	informace o spojení	P
T	čas počátku a konce relace	P
I	informace o relaci	V
U	URI	V
E	email	V
P	tel. číslo	V
B	informace o šířce pásma	V
R	počet opakování	V
Z	korekce časových zón	V
K	šifrovací klíč	V
A	řádky s atributy	V
M	Informace o multimédiích	V
a	atributy multimédií	V

2.1.2 ACK

Zprávy ACK se používají jako potvrzení poslední odpovědi na žádost INVITE. Odpovědi na ostatní žádosti se nepotvrzují. Zpráva opět obsahuje úvodní metodu ACK a pole Call–ID, CSeq (které není nikdy pro ACK inkrementováno), From, To, Via a Max–Forwards. Ty jsou naplněny informacemi právě z odpovědi, kterou potvrzují.

Příklad zprávy ACK:

```
ACK sip:marconi@tower.radio.org SIP/2.0
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bK321g
Max-Forwards: 70
To: G.Marconi <sip:marconi@radio.org>;tag=a53e42
From: NikolaTesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: 123456789@lab.high-voltage.org
CSeq: 1 ACK
Content-Length: 0
```

2.1.3 BYE

Zprávu BYE může vyslat kterákoliv strana (pouze UA, nikoli proxy či jiný server) probíhajícího hovoru a oznamuje druhé straně ukončení přenosu. Hovor je považován za probíhající, pokud byla na zprávu INVITE odeslána kladná odpověď, která byla následně potvrzena pomocí ACK. BYE obsahuje opět název metody a povinné hlavičky Call–Id, CSeq, From, To, Via a Max–Forwards, obdobně jako je tomu ve zprávě INVITE. Číselná hodnota v poli CSeq již může být jiná, než tomu bylo v předchozích případech, protože se již jedná o jinou transakci, jinou část dialogu.

Konstrukce zprávy je podobná zprávě INVITE, pouze metoda je tentokrát BYE a obě pole To a From obsahují tagy příslušných stran.

Zpráva BYE pak může vypadat takto:

```
BYE sip: n.tesla@lab.high-voltage.org SIP/2.0
Via: SIP/2.0/UDP tower.radio.org:5060;branch=z9hG4bK392kf
Max-Forwards: 70
To: NikolaTesla <sip:n.tesla@high-voltage.org>;tag=76341
From: G.Marconi <sip:marconi@radio.org>;tag=a53e42
Call-ID:1 23456789@lab.high-voltage.org
CSeq: 1 BYE
Content-Length: 0
```

2.1.4 CANCEL

Tato zpráva je podobná poslední zmiňované BYE, ale používá se v jiném případě a má také trochu jinou strukturu. Zatímco BYE oznamuje ukončení již probíhajícího hovoru, zpráva CANCEL se zasílá pro zrušení navazovaného spojení. Typicky se tato zpráva využívá při vyhledávání protější strany nebo v situaci, kdy volaný účastník nepřijímá hovor. Dále, když proxy server vyhledává volanou stranu více cestami a z některé z nich se vrátí zpráva o doručení, je třeba zrušit ostatní požadavky. Z posledního jmenovaného příkladu je zřejmé, že CANCEL není zpráva mezi koncovými body sítě, ale patří pouze nejbližší entitě, například statefull proxy serveru. Ten pak generuje vlastní CANCEL zprávu a zasílá ji tam, kam on sám potřebuje. Podobně jako ACK, ani tato metoda nemění hodnotu v poli CSeq, aby byla zrušena patřičná zpráva INVITE. Také pouze tuto žádost má smysl rušit, protože jako jediná může trvat déle, než se spojení naváže. Na ostatní zprávy musí příjemce reagovat okamžitě, jinak by došlo k tomu, že bude zpráva CANCEL generována dříve, než dorazí finální odpověď na INVITE. Tato situace může nastat u proxy serveru, který obdrží zrušení volání od UA1, okamžitě na tuto zprávu odpovídá, a teprve poté jí přeposílá dále. Pokud mu mezitím přijde od UA2 potvrzení INVITE zprávy, proxy server ji přepošle UA1. Ten následně musí generovat novou zprávu BYE pro regulérní ukončení spojení.

Složení zprávy CANCEL je následující:

```
CANCEL sip:i.newton@cambridge.edu.gb SIP/2.0
Via: SIP/2.0/UDP 10.downing.gb:5060;branch=z9hG4bK3134134
Max-Forwards: 70
To: IsaacNewton<sip:i.newton@cambridge.edu.gb>
From: ReneDescartes<sip:visitor@10.downing.gb>;tag=034323
Call-ID: 42@10.downing.gb
CSeq: 32156CANCEL
Content-Length: 0
```

2.1.5 REGISTER

Zpráva k registraci u registrar serveru obsahuje, kromě již zmíněných povinných polí, navíc povinné pole Contact, které se může vyskytovat několikrát. Kromě samotného URI uživatele může další hlavička obsahovat klíčové slovo „mailto:“ a emailovou adresu, na kterou je možné odeslat zprávu. Hodnoty ve zmíněných polích

však mají mírně odlišný význam. Pole From a To běžně obsahují stejné URI určené k registraci. From má význam původce žádosti, naproti tomu To určuje, kdo má být zaregistrován, protože v SIP je podporováno registrování třetí strany. Call–ID je pak použito stejné pro všechny žádosti o registraci. Důvod registrace je zřejmý – je potřeba se za proxy serverem identifikovat, aby bylo možné UA dohledat, ať již pomocí URI, nebo pole Contact. Další nepovinnou hlavičkou ve zprávě REGISTER je pole Expires, které udává časovou platnost záznamu u registrar serveru. V RFC jsou deklarovány speciální hodnoty polí Contact a Expires, na které server patřičně reaguje. Tato pole však nejsou podstatná, a proto nebudou dále rozebírána.

Metoda Register zahajuje novou transakci uvnitř dialogu, hodnota pole CSeq se tedy zvyšuje. Její podoba je uvedena níže:

```
REGISTER sip:registrar.athens.gr SIP/2.0
Via: SIP/2.0/UDP 201.202.203.204:5060;branch=z9hG4bK313
Max-Forwards: 70
To: sip:euclid@athens.gr
From: <sip:secretary@academy.athens.gr>;tag=543131
Call-ID: 2000-July-07-23:59:59.1234@201.202.203.204
CSeq: 1 REGISTER
Contact: sip:euclid@parthenon.athens.gr
Contact: mailto:euclid@geometry.org
Content-Length: 0
```

2.1.6 OPTIONS

Zpráva OPTIONS je generována pouze UA za účelem zjištění schopností jiné entity v síti. Zpráva se opět skládá z metody a ze stejných povinných polí, která jsou ve zprávě INVITE. I odpověď entity je obdobná, jako na zprávu INVITE. Negativní odpověď je klasická, ale kladná tentokrát může obsahovat pole Allow, Accept, Accept–Encoding, Accept–Language a Supported. Význam jednotlivých polí lze najít přímo v RFC dokumentu.

Obsah této zprávy je závislý od jejího použití, ale pro názornost lze uvést například tuto:

```
OPTIONS sip:user@carrier.com SIP/2.0
Via: SIP/2.0/UDP cave.kings.cambridge.edu.uk
;branch=z9hG4bK1834
Max-Forwards: 70
```

```
To: <sip:user@proxy.carrier.com>
From: sip:james.maxwell@kings.cambridge.edu.uk;tag=34
Call-ID: 9352812@cave.kings.cambridge.edu.uk
CSeq: 1 OPTIONS
```

2.1.7 Ostatní zprávy

Pro použití v SIP jsou definovány i další žádosti, kterým jsou věnovány jiné RFC dokumenty, než je samotný RFC 3261 ([1]). Pro zajímavost to jsou například:

- REFER (RFC 3515)
- NOTIFY (RFC 3265)
- MESSAGE (RFC 3428)
- PRACK (RFC 3262)
- UPDATE (RFC 3311)
- INFO (RFC 2976).

2.2 SIP odpovědi

SIP responses, neboli odpovědi v SIP, jsou generovány výhradně serverovou částí UA (označované jako UAS, proti klientské části UAC) a v běžném textu jsou označovány kódem odpovědi a textovým popisem. Nejběžnější jsou 200 OK, 180 RINGING, 100 TRYING a další. Až na výjimky mají zprávy podobný obsah jako žádosti – hlavičky a hodnoty. Místo metody však obsahují číselný kód, textový popis a často také tělo zprávy. Odpovědi se dělí do tříd, které jednoznačně, avšak velmi obecně definují význam zprávy. Zamezí se tak případným komplikacím při obdržení nedefinované zprávy a její mylné interpretaci či dokonce nepochopení. Pokud například entita nezná zprávu s číslem 483, pak ji musí vyhodnotit podle třídy 4 a podle toho také zareagovat.

Třídy byly vybrány podle HTML s několika málo rozdíly. Tedy ne všechny HTML odpovědi jsou kompatibilní se SIP a naopak. Textové popisy odpovědí byly upraveny pro použití v SIP a některé nové zprávy byly definovány nad rámec HTML. Následující seznam stručně popisuje jednotlivé třídy odpovědí:

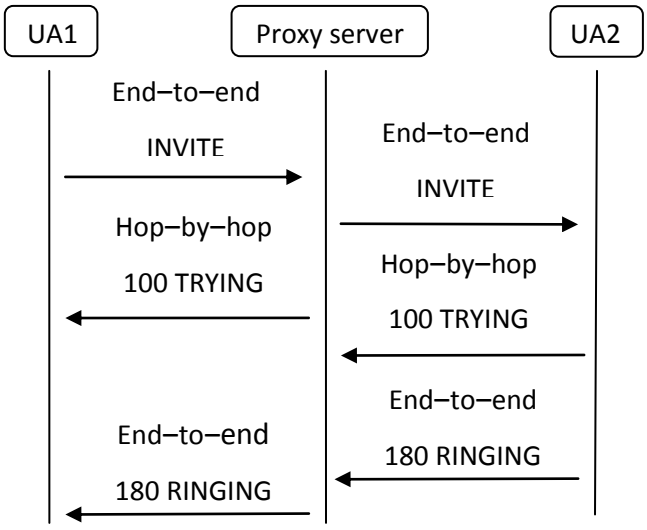
- 1** Information – oznámení o zpracovávání žádosti
- 2** Success – kladná odpověď na zaslaný požadavek
- 3** Redirection – přesměrování
- 4** Client error – chyba na straně klienta
- 5** Server error – chyba na straně serveru
- 6** General error – obecná chyba

Symboly ** jsou nahrazeny dílčími číselnými kombinacemi zpřesňujícími význam zprávy. Nově definované zprávy, jsou kvůli předcházení kolizí číslovány až od 80.

Textové popisy, které doprovází kód zprávy, mají pouze informativní charakter pro uživatele. Entity se vždy rozhodují podle číselného kódu. Tedy pokud UA přijme zprávu 200 CANCELLED (namísto 200 OK), zpracuje ji jako kladnou odpověď na příslušnou žádost. Popis tedy může být libovolný, a jelikož je také koncovému uživateli zobrazován, lze jej využít například k „polidštění" systémových zpráv nebo pro detailní popis u nově definovaných zpráv. Mnoho typů odpovědí je definováno v RFC dokumentech, zde je uvedeno pouze několik příkladů k jednotlivým třídám.

2.2.1 Information

Jak již bylo zmíněno, zprávy první třídy začínají číslem 1 a mají informační charakter. Nejčastěji podávají zprávu o průběhu vytváření spojení a potvrzují úspěšné přijetí požadavku, ale nic nevypovídají o tom, zda bude žádosti vyhověno, či nikoli. Vždy jsou typu end–to–end – přeposílají se od jedné koncové entity k druhé. Jedinou výjimkou je zpráva 100 TRYING, která je typu hop–by–hop, tedy generuje se pouze mezi dvěma sousedními entitami. Příklad těchto typů zpráv je uveden na Obr. 6. Zprávy prvního typu jsou vždy přeposílány v jasném časovém sledu, kdežto druhý typ zpráv se vždy řídí příjemcem. Na zmíněném obrázku proxy server přijme zprávu INVITE a žadateli odpoví 100 TRYING po přeposlání INVITE. Když i od druhé koncové stanice přijme tutéž odpověď s kódem 100, tak ji nikam dále nepřeposílá a zpracovává ji sám – například pro zastavení rozesílání žádosti INVITE do různých částí sítě.



Obr. 6 – End–to–end a hop–by–hop typy zpráv

Vzhledem k možné větší časové náročnosti vyhledávání cílového UA je vhodné, aby proxy server odeslal tuto „prozatímní“ odpověď, díky čemuž by vysílající UA měl přehled o stavu a nedošlo by k vypršení kontrolních časovačů. Odpovědi 100 také neobsahují tělo zprávy, protože ztrácí význam rozšíření významu kódu.

Dalším příkladem zprávy první třídy je již zmiňovaná 180 RINGING, oznamující úspěšné přijetí INVITE a vyvolávání uživatele. Tato odpověď nemusí být odeslána, pokud bude hovor okamžitě přijat a UAS odesílá rovnou 200 OK. Podobně i u zpráv 181 CALL FORWARDED a 182 CALL QUEUED obsahující i počet účastníků ve frontě.

2.2.2 Success

Zprávy začínající číslem 2 nepotvrzují pouze úspěšné přijetí žádosti jako předchozí třída, ale oznamují její kladné vyřízení. Nejběžněji používanou odpovědí je 200 OK, která souhlasí s požadavkem na spojení (žádost INVITE), a tedy obsahuje i tělo zprávy s popisem přenášených dat. Stejně tak obsahuje tělo i odpověď na žádost OPTIONS popisující schopnosti druhé strany. Odpověď na ostatní žádosti tělo neobsahuje a pouze oznamuje vyhovění zaslanému požadavku.

Zpráva 200 OK, odesílaná na zprávu INVITE z předchozí kapitoly:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
;received=100.101.102.103
To: G.Marconi <sip:marconi@radio.org>;tag=a53e42
From: NikolaTesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: 123456789@lab.high-voltage.org
CSeq: 1 INVITE
Contact: <sip:marconi@tower.radio.org>
Content-Type: application/sdp
Content-Length: 155
v=0
o=Marconi2890844528289084452 8INIP4tower.radio.org
s=PhoneCall
c=INIP4200.201.202.203
t=00
m=audio60000RTP/AVP0
a=rtpmap:0PCMU/8000
```

2.2.3 Redirection

Odpověďmi 3** oznamuje proxy server, že hledaný UA je přesunut do jiné části sítě. Obsahují hlavičku Contact, ve které je uvedeno nové URI, které má entita vyhledávat. Význam zpráv 301 MOVED PERMANENTLY a 302 MOVED TERPORARILY je zřejmý. K druhé zmíněné zprávě je vhodné poznamenat, že může obsahovat hlavičku Expires, jenž vyznačuje delší dobu trvání přesunutí hledané entity. Další příklad je zpráva 300 MULTIPLE CHOICES, která obsahuje několik hlaviček Contact s novou pozicí volaného. Pořadí těchto hlaviček je dáno předpokládaným výskytem volaného UA. Zpráva 380 ALTERNATIVE SERVICE se použije v situaci, kdy hledaný UA vyžaduje jinou službu, jako například přesměrování do hlasové schránky.

2.2.4 Client error

Třída zpráv oznamující nemožnost zpracování požadavku z důvodu chyby jeho odesílatele. Takový požadavek pak nesmí být odeslán znovu bez úprav nebo jinému příjemci. Konkrétní odpovědi specifikují, o jakou chybu se jedná.

400 BAD REQUEST indikuje nedefinovanou metodu v úvodu žádosti, chybějící hlavičky (To, CSeq apod.), nebo pokud UAS přijme několik různých zpráv INVITE určených pro stejné Call–ID. Zprávy s označením 401 UNAUTHORIZED (407 PROXY AUTHENTIFICATION REQUIRED) zasílá UAC (UAS) v případě, že je požadována autorizace uživatele. Zpráva obsahuje zvláštní hlavičku s informacemi o možném způsobu autorizace a UAC má možnost znovu zaslat doplněnou žádost. Obdobně registrar server odpovídá zprávou 401 na žádost REGISTER, která neobsahuje potřebné údaje. Znovu posílaná žádost by měla obsahovat shodné Call–ID, aby ji bylo možné spojit s předchozím neúspěšným pokusem. Při zamítnutí autorizace je odeslána odpověď 403 FORBIDDEN. Namísto autorizace může být požadována platba indikovaná zprávou 402 PAYMENT REQUIRED. Chyba 404, známá z HTML, značí nenalezení žádaného UA, 480 TERPORARILY UNAVAILIBLE pak jeho dočasnou nedostupnost.

Chybových zpráv je definováno velké množství pro mnoho specifických situací, cílem této části je pouze stručný přehled každé třídy SIP odpovědí, nikoliv detailní rozbor. Seznam s detailním popisem všech definovaných zpráv lze najít například v RFC 3261 ([1]) a dalších rozšiřujících dokumentech.

Příklad odpovědi s kódem 407:

```
SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/UDP discrete.sampling.org:5060;branch=z9hG4bK6563
;received=65.64.140.198
From: Shannon <sip:shannon@sampling.org>;tag=59204
To: Schockley<sip:shockley@transistor.com>;tag=142334
Call-ID: adf8gasdd7fld@discrete.sampling.org
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="sampling.org",qop="auth",
nonce="9c8e88df84df1cec4341ae6cbe5a359",
opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

2.2.5 Server error

Hlášení o chybě na straně serveru začínají číslem 5, často obsahují hlavičku Retry–after. Ta obsahuje informaci, za jak dlouho je možné požadavek opakovat, pokud server předpokládá, že je problém dočasný. Požadavek je taktéž možné směrovat na jinou entitu.

Zpráva 500 SERVER INTERNAL ERROR je odesílána, pokud server není schopen zpracovat požadavek z příčin uvedených uvnitř zprávy. Příčina vyslání zprávy 501 NOT IMPLEMENTED je zřejmá, stejně jako 503 SERVICE UNAVAILABLE, 505 VERSION NOT SUPPORTED. Zpráva 502 BAD GATEWAY je generována serverem, který plní funkci hraniční brány a zaznamená problém v předání požadavku do druhé sítě.

2.2.6 General error

Zprávy s kódem 6** jsou používány v situacích, kdy je server schopen rozpoznat, že žádost nebude zpracována ani v jiných částech sítě – tedy není žádoucí její přeposílání jiným entitám sítě. Zprávy opět mohou obsahovat pole s časovým údajem, po kterém bude server pravděpodobně schopen požadavek obsloužit. Odpověď 600 BUSY EVERYWHERE je obdobou zprávy 486 BUSY HERE, generovanou UAS, v tomto případě je ale server informován, že žádost nemůže být obsloužena ani na jiných místech v síti. Kód 603 DECLINE je zaslán, pokud volaná strana spojení nepřijímá. 604 DOES NOT EXIST ANYWHERE je obdobou zprávy 404, avšak indikuje navíc skutečnost, že hledané URI není k nalezení v celé obsluhované SIP síti.

3 SIP knihovna

Další kapitola této práce se bude zabývat již praktickou částí a to SIP klientem (UA). Nejprve bude nutné vybrat programovací jazyk, ve kterém bude aplikace psána. To pak ovlivní i výběr knihoven. Použití již napsaných a volně dostupných knihoven má výhodu v menší časové náročnosti, oproti vytvoření knihoven vlastních. Hotové knihovny jsou sice robustnější a obsahují více možností, než by bylo potřeba pro dosažení cíle této práce, ale psaní knihoven od základu by zabralo příliš mnoho času. Významné je také odzkoušení komunitou, jeden samotný autor by velmi obtížně odhaloval všechny chyby. Některé hotové knihovny (anglicky stack) obsahují metody a třídy jak pro SIP, tak pro SDP či RTP. Některé jsou zaměřeny pouze na jeden z protokolů a další navazující funkcionality je třeba řešit další samostatnou knihovnou.

Programovací jazyk byl zvolen C++, pro svou jednoduchost oproti C, stabilnost a rychlost proti jazyku Java. Protože se práce bude zabývat klientem pro operační systém Microsoft Windows, i vývojové prostředí bude zvoleno od stejné firmy a to MS Visual Studio.

Výběr samotné knihovny však není tak jednoduchý, protože je na internetu dostupné větší množství horších či lepších řešení. Některé knihovny zde budou stručně popsány a budou zváženy jejich výhody a nevýhody.

3.1 oSIP

Jedná se o knihovnu pro operační systémy Windows, Linux a iOS, vyvíjenou v jazyce C. Dnes je dostupná již druhá verze oSIP2, která vychází z první – nejedná se tedy o kompletně nový produkt. Výhodou je licence L–GPL (*Lesser General Public Licence),* která umožňuje její komerční využití bez zveřejnění zdrojových kódů aplikace. Také obsahuje dostatečnou dokumentaci.

Další výhodou může být pro některé situace i přidružený software. Všechny zmíněné produkty jsou distribuovány pod licenci GPL. Příkladem je knihovna *eXosip*, která přináší rozšíření standardního SIP a snazší implementaci vyšších vrstev. *Partysip* implementuje proxy server s možností registrace, autentifikace a směrování. A nakonec *linphone* je webová telefonní aplikace.

Její nevýhodou je pak použití jazyka, který je o mnoho složitější než vybrané C++, a proto by práce s touto knihovnou byla časově náročnější.

3.2 reSIProcate.org

Tato knihovna je psána v C++ a také pro více operačních systémů (kromě Windows), je objektově orientovaná a pod velmi liberální VOVIDIA licencí, která se skládá z mnoha částí, týkajících se pouze určitých částí knihovny. Její vývoj byl zřejmě na čas přerušen, poslední verze je z prosince roku 2009. Obsahuje velké množství funkcí: softwarového klienta pro uskutečňování hovorů, proxy a registrar server, IM server a mnoho dalších.

3.3 SIP foundry

Tato komunita vyvíjí velké množství nástrojů pro práci se SIP, ale i jinými protokoly. Knihovna právě pro SIP je napsaná pro C++ a pod licencí L–GPL, obsahuje opět části pro UA, stateless proxy server i pro registrar. Pro multimediální přenos používá RTP, ale tato část je psána pro jazyk JAVA. Také je dostupná obsáhlá dokumentace a fórum pro řešení chyb.

3.4 Sofia–SIP

Tato knihovna je zaměřena pouze na UA, je přehlednější oproti ostatním knihovnám. Také je velmi pozitivně hodnocena programátory a všeobecně doporučována pro tvorbu vlastního SIP klienta. Je primárně zaměřena pro Linux (ostatní platformy však nejsou překážkou) a licencována je pod LGPL. Jazykem této knihovny je C. Taktéž je v knihovně obsažena část pro multimediální přenos. Poslední verze je však z prosince roku 2008.

3.5 OPAL a PTlib

Knihovny OPAL a PTlib jsou vyvíjeny komunitou a společně jsou použity pro sestavení softwarového telefonu OpalPhone. Poslední finální a stabilní verze pochází z počátku roku 2011 a na další se již pracuje. Obsahuje rozsáhlou dokumentaci samotných knihoven, avšak neobsahuje návody na práci s knihovnou. Ke stažení jsou také pouze zdrojové kódy, a je tedy třeba je sestavit dle podrobného návodu. Knihovna OPAL zahrnuje jak protokol SIP, IAX a H.323, tak i SDP a RTP. Implementuje také kodeky pro audio (G.711 a GSM06.10) i video (H.261 a H.263). PTlib pak obstarává použití napříč platformami, a to včetně mobilní. Tyto knihovny nakonec byly vybrány pro svou komplexnost, robustnost a také kvůli velmi rozsáhlé dokumentaci. Projekty pro kompilaci obsahují také několik vzorových příkladů použití, a to jak ve formě velmi

jednoduché konzolové aplikace, tak i ve formě grafické aplikace s nepřeberným množstvím funkcí a rozmanitou paletou zobrazovaných parametrů a detailů. Částečně je také zmíněno její použití v [4], nicméně nijak podrobně.

Jak se později ukázalo, právě forma zdrojových kódů a jejich nutná kompilace přináší velké množství problémů. Velmi rozsáhlá dokumentace sice vystupuje jako mocný nástroj, leč nepřesné nebo úplně chybějící tutoriály byly hlavními faktory neúměrné časové náročnosti vývoje aplikace.

4 Praktická část

Následující kapitola se bude zabývat vývojem aplikace, která implementuje zvolený signalizační protokol při využití vybraných knihoven. V první části bude popsán postup sestavení knihoven PTlib a Opal a také řešení několika problémů, které se sestavením souvisí. Další podkapitoly pak popíší různé cesty, kterými se vývoj aplikace ubíral. Na závěr bude popsán samotný návrh a realizace aplikace.

4.1 Příprava knihoven

Jak již bylo popsáno, knihovny jsou dostupné ke stažení ve formě zdrojových kódů a je třeba je sestavit na platformě, pro kterou mají být použity. Pro tento úkon jsou na stránkách vývojářů připraveny návody jak pro platformu Windows, tak pro linuxové systémy.

Bohužel se návody, jak pro knihovnu OPAL, tak i pro PTlib, ukázaly jako nedostačující. Kroky lze bez potíží sledovat, výsledky se však nedostavují podle představ. Zejména u knihovny PTlib, kterou je nutné zkompilovat jako první, se objevuje problémů nejvíce. Jeden z kroků původního návodu je označen jako volitelný a obsahuje seznam přídavných balíčků s jejich použitím podle požadavků na budoucí využití knihoven. Některé části tohoto kroku jsou ale bohužel nutné pro sestavení vzorových příkladů, o kterých se ovšem návod vůbec nezmiňuje.

Také nejsou nijak zmíněna specifika konkrétních verzí operačního systému, a proto i jim je třeba postup uzpůsobit. Počínaje Windows Vista se oproti předchozím verzím OS firmy Microsoft výrazně změnila bezpečnostní politika uživatelských účtů, zejména v oblasti práv přístupu k některým částem systému. Při kompilaci knihoven je nutná i práce s některými soubory, ke kterým nemá přístup ani uživatel s rozšířenými právy správce systému. Je tedy nutné celé vývojové prostředí spouštět pod účtem Administrátora. Bohužel při neúspěšné kompilaci, která selže právě z tohoto důvodu, nejsou vypisované chyby signifikantní a jejich odstranění tak není na první pohled zřejmé.

V průběhu vzniku knihoven procházelo vývojem také prostředí, ve kterém (na platformě Windows) měly být knihovny zpracovávány. Návody [5] tak pracují s několika verzemi zdrojových projektů určených ke kompilaci, ovšem zpětná kompatibilita nejnovějších řešení prozatím zůstala pouze na teoretické úrovni. Uživatel

tedy mohl podlehnout dojmu, že zpracovat zdrojové kódy pod nejnovější verzí Microsoft Visual Studia bude bezproblémové. Ve skutečnosti však příslušné soubory prozatím chyběly a konverze starších řešení neproběhla tak, jak měla. Bylo proto nutné celé prostředí odstranit ze systému a obstarat vývojové prostředí v nižší verzi. Spolu se starší verzí Visual Studia, bylo později nezbytné také doinstalovat balík aktualizací, spolu se servisním balíčkem, který kromě opravy chyb přinášel i drobné nové funkcionality prostředí. Bohužel již nebylo zřejmé, zda nové funkce, či opravené chyby pomohly odstranit některé z vyskytujících se problémů.

Po úspěšném sestavení knihoven přišlo na řadu zpracování vzorových příkladů. U knihovny PTlib měl obsažený program *Hello World* prezentovat pouze portabilitu knihovny. Na žádané platformě Windows funkční byl, ovšem více nepřinášel. Oproti tomu u knihovny Opal byly příklady o mnoho zajímavější.

První ze dvou byl *simpleOpal*, který představoval konzolovou aplikaci, která se ovládala pouze pomocí příkazů a parametrů při spuštění. Kompilace tohoto příkladu byla vzhledem k jeho jednoduchosti a nenáročnosti bezproblémová.

Druhý příklad, nesoucí název *OpenPhone*, byl naopak velmi komplikovaný a vyžadoval právě mnoho dodatečných kroků pro správnou kompilaci. Ovšem návody tvůrců o tomto nehovoří a ani komunita ostatních vývojářů své zkušenosti neprezentovala. Jedním z prvků, který OpenPhone pro svůj běh používá je jazyk XML. Vývojové prostředí však nemá vhodné nástroje pro zpracování XML proudů a je tedy nutné z volitelných balíčků doinstalovat Expat XML parser, který se o to postaral. Ten je nabízen ve formě instalačního souboru i v komprimované podobě bez potřeby instalace v souboru *.tar.gz. Pokud by vývojář zvolil druhou možnost, sice se vyhne instalaci nástroje do systému, ale bude vystaven riziku nesprávné dekomprimace. Při rozbalování souboru totiž některé archivační nástroje mohou špatně reprezentovat konec řádku nebo konec souboru. Při kompilaci pak vznikají chyby, které opět nenesou jednoznačnou informaci o tom, kde lze chybu nalézt a odstranit. Naštěstí se v tomto konkrétním případě jednalo o chybu jediného textového souboru, jenž bylo možné ručně editovat a odstranit špatně zapsaný konec souboru a chybu tak opravit. Pokud ovšem není specifický důvod pro komprimovaný archiv, je vhodnější použití instalačního balíčku.

Celé uživatelské rozhraní OpenPhone je postaveno na nástroji wxWidgets, které přináší velké množství funkcí a umožňuje realizaci široké palety prezentačních nástrojů, týkajících se samotné aplikace, probíhajících hovorů a dalších parametrů. Také zde je nutné doinstalovat další nástroj a provést úpravy v operačním systému, spočívající v přidání globální proměnné. Bohužel i v této části se objevovaly problémy spojené se správnou implementací tohoto nástroje do systému a vývojového prostředí. Poslední stabilní verze 2.9 byla vedena již jako nová řada a prošla několika změnami. Ovšem nezachovávala 100% zpětnou kompatibilitu s předchozími verzemi a tudíž ji nebylo možné použít pro kompilaci vzorového příkladu. Poslední finální verze předchozí řady byla tou správnou, pro použití ke kompilaci daných verzí knihoven.

4.2 Sestavení knihoven

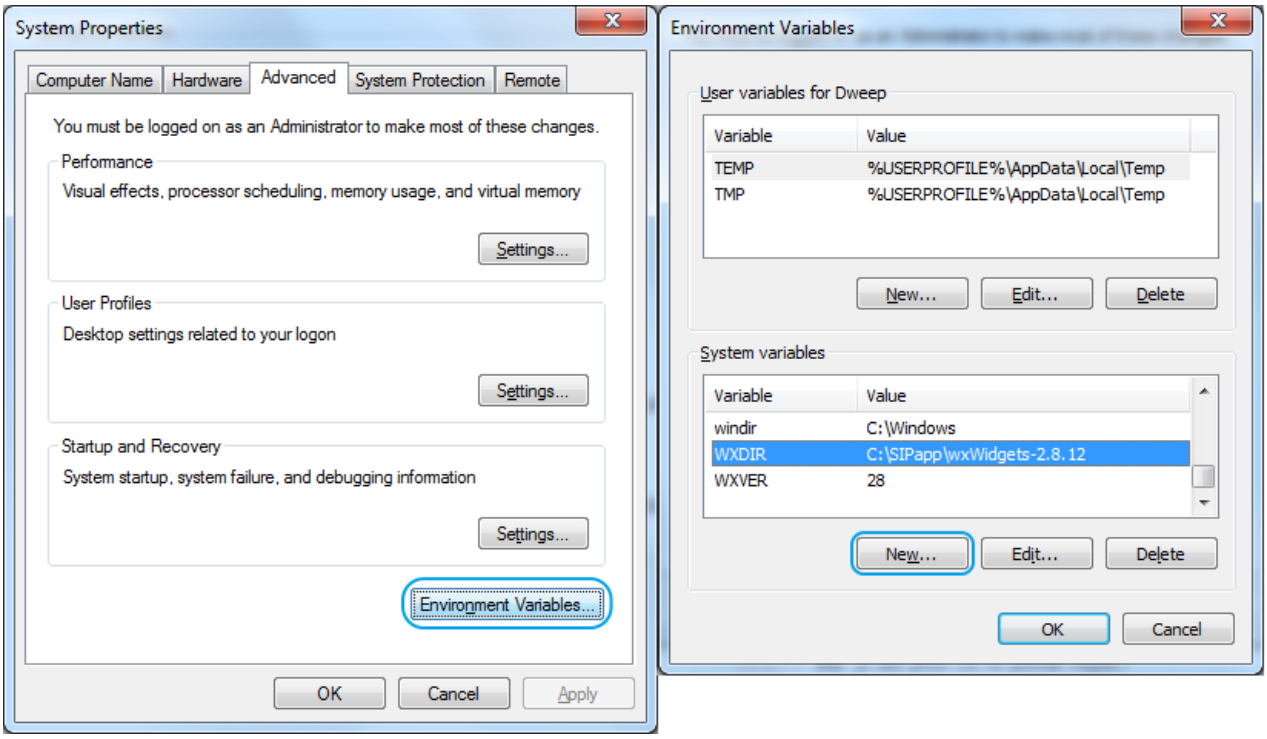
Nejprve je potřeba definovat v jakém prostředí a s jakými verzemi bylo pracováno. Obecně je nutné konstatovat, že použití správných verzí všech prvků bylo klíčové, protože stále probíhá vývoj samotných knihoven i nástrojů, které jsou potřeba k jejich kompilaci. K práci bylo tedy použito:

- Microsoft Visual Studio Professional Edition a to ve verzi 2008; Jak bylo zmíněno, verze 2010 by měla být podporována, nicméně potřebné soubory pro otevření chybí a při převodu projektu na vyšší verzi dochází k chybám.
- K Visual Studiu příslušný .NET Framework ve verzi 3.1 a vyšší
- Knihovny s kódovým označením Sirius a konkrétně Opal 3.8.4 a PTlib 2.8.4
- Nástroj GNU Bison, který je předpřipravený ke stažení přímo v návodu ([5])
- XML parser Expat byl použit ve verzi 2.0.1
- Pro grafické rozhraní vzorového příkladu OpenPhone nástroj wxWidgets ve finální verzi předposlední řady – 2.8.12

V neposlední řadě je nutné uvést, že vývoj byl veden pod 64-bitovým OS MS Windows 7 Professional.

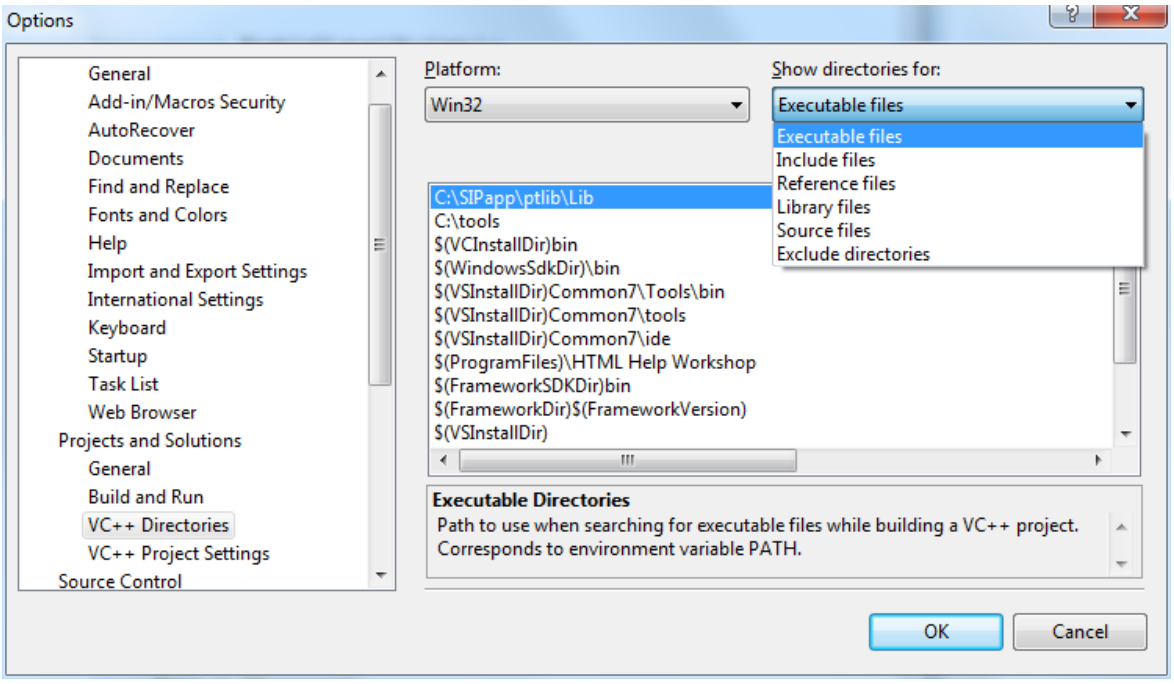
Na základě dříve zmíněných komplikací zde bude popsán upravený postup k sestavení obou knihoven a zprovoznění příkladů pro uvedenou platformu. V následujícím textu bude použit anglický výraz *solution*, jehož český překlad by mohl být zavádějící. Solution označuje soubor pro Visual Studio, který zastřešuje několik projektů a uchovává globální nastavení společné pro všechny projekty.

1. Prvním krokem je samozřejmě instalace vývojového prostředí a aktualizace OS kvůli bezpečnostním záplatám a opravám chyb v software. Důležitým krokem se také ukázala instalace servisního balíku Microsoft Visual Studio 2008 SP1, opravujícího některé chyby, které při překladu vznikají.
2. Dále je na systémovém disku potřeba v kořenovém adresáři vytvořit adresář „tools“, do kterého se rozbalí zmíněný nástroj GNU Bison – jedná se o soubory *flex.exe* a *bison.exe*, adresář *share* obsahující soubory *bison.simple* a *bison.hairy*.
3. Následujícím krokem je instalace wxWidgets pomocí staženého instalátoru a stejným způsobem instalace i XML parseru Expat. Pro použití wxWidgets je navíc potřeba v operačním systému zavést novou proměnnou. To se provede přes *Pokročilé nastavení systému [Advanced system settings]* (pravým tlačítkem na *Počítač [Computer]* → *Vlastnosti [Properties]*) a na nově otevřeném okně přes *Proměnné prostředí [Environment Variables]*. V dalším novém okně se ve spodní části (označené *Systémové proměnné [System Variables]*) pomocí tlačítka *Nová [New]* přidá proměnná WXDIR s hodnotou nastavenou na cestu do adresáře, kam byly instalovány wxWidgets a proměnná WXVER s hodnotou 28 (dle verze). Viz Obr. 7.



Obr. 7 – Zavedení nové systémové proměnné v anglickém prostředí Windows 7

4. Knihovny PTlib a Opal lze rozbalit do libovolného adresáře, který bude v této práci označován jako *InstallDir* (například, jsou-li knihovny rozbaleny do *C:/SIPapp, bude cesta k adresářům knihovny Opal označena jako InstallDir/opal* – ve skutečnosti se bude jednat o *C:/SIPapp/opal*).
5. Dále je možné přikročit ke spuštění VS, které je ale nutné spustit s právy administrátora. Přichází na řadu důležité nastavení samotného VS. V menu *Tools*/Options se otevře dialogové okno (viz Obr. 8) a v něm je třeba vybrat z levého menu položku *Projects and Solutions/VC++ Directories* a zde se budou přidávat cesty pro jednotlivé soubory. Pro *Executable files* musí být přidána cesta *C:/tools* a *InstallDir/ptlib/Lib*, pro *Include files* cesty *InstallDir/opal/include* a *InstallDir/ptlib/include*, *Library files* cesty *InstallDir/opal/lib* a *InstallDir/ptlib/Lib*. Toto nastavení je velmi důležité, aby překladač dokázal nalézt potřebné soubory při zpracování.



Obr. 8 – Nastavení cest ve Visual Studiu 2008

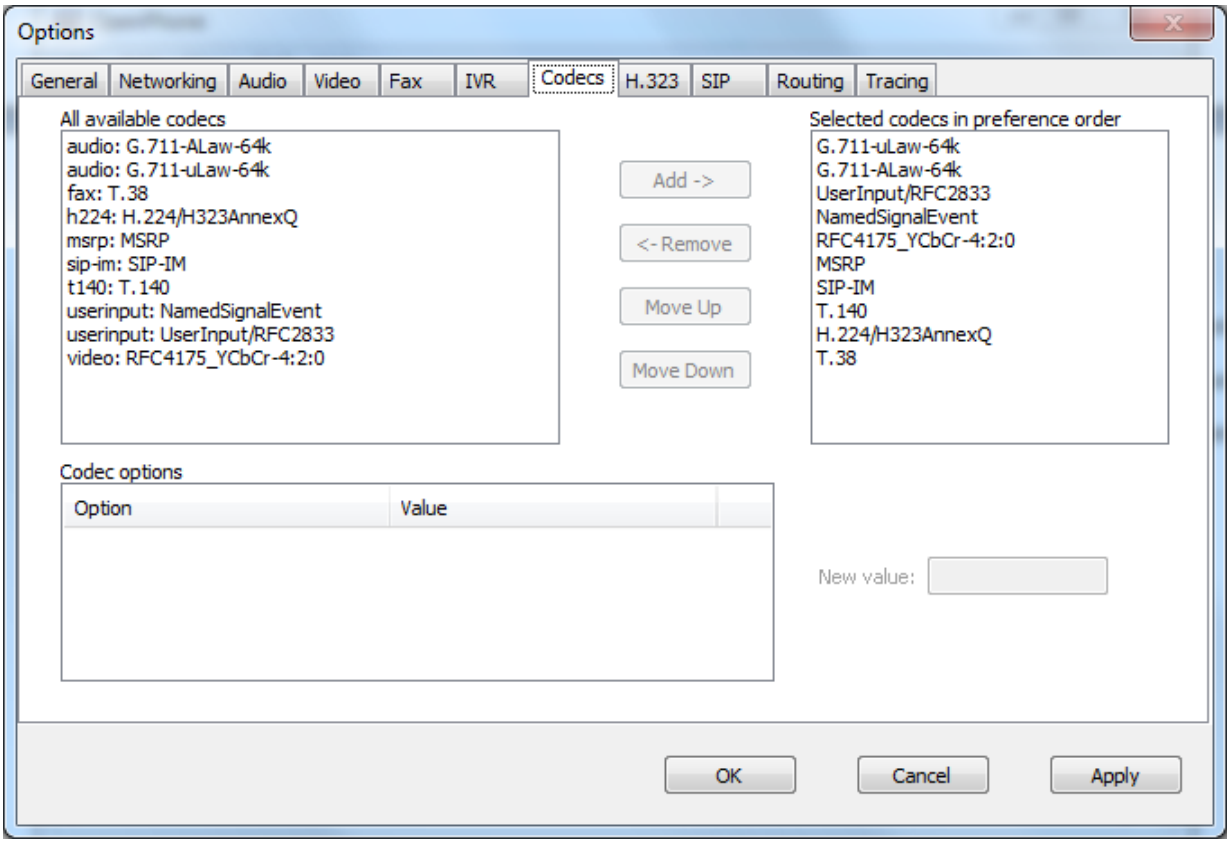
6. Předposledním přípravným krokem je sestavení knihoven pro wxWidgets – v instalačním adresáři a podadresářích *build/msw* se nachází soubor wx.dsw, který otevře solution ve Visual Studiu. Kvůli kompatibilitě se starší verzí Visual Studia je potřeba provést konverzi jednotlivých projektů. Přes hlavní menu se pak celé solution sestaví a to v režimu Unicode Release, i Unicode Debug. Následně se celé solution uzavře a stejným způsobem se sestaví projekty ze souboru *utils/wxrc/wxrc.dsw*. Na závěr je nutností nakopírovat soubor *wxrc.exe* z adresáře *utils/wxrc/vc_mswu* do ručně vytvořeného adresáře *bin* ve stejném instalačním adresáři.
7. Závěr přípravy spočívá v sestavení nástroje MergeSym. Ten lze vytvořit ve stejnojmenném projektu v solution *ptlib_2008.sln,* a to jak v režimu Release, tak i Debug. Následně je nutné přesunout soubor *MergeSym.exe* z adresáře *InstallDir/lib/MergeSym/Debug* do *C:/tools*.
8. Po tomto nastavení je na řadě již samotné sestavení knihoven. Je nutné dodržet pořadí, kvůli závislosti projektů jednoho na druhém. Jako první proběhne sestavení projektu *ptlib_static*, v obou režimech (Debug a Release) a obdobně i *ptlib_dll*. Po bezproblémové kompilaci těchto projektů je možné otevřít solution *opal_2008.sln*, ve kterém se stejným postupem sestaví nejprve *opal_static* a následně *opal_dll*.
9. Na závěr je již možné otevřít i solution *ptlib_samples_2008.sln* a *opal_samples_2008.sln* a přeložit vzorové příklady.

4.3 Vývoj aplikace

Vzhledem k tomu, že kompilaci knihoven provázelo velké množství problémů, jejichž řešení bylo časově poměrně náročné, bylo upuštěno od původního plánu vývoje kompletně nové, samostatné aplikace. Nadále se vývoj soustředil na využití již existujících řešení, jejich doplnění, zjednodušení a další úpravy. Tato volba slibovala menší zatížení na samotný vývoj a také rychlejší postup.

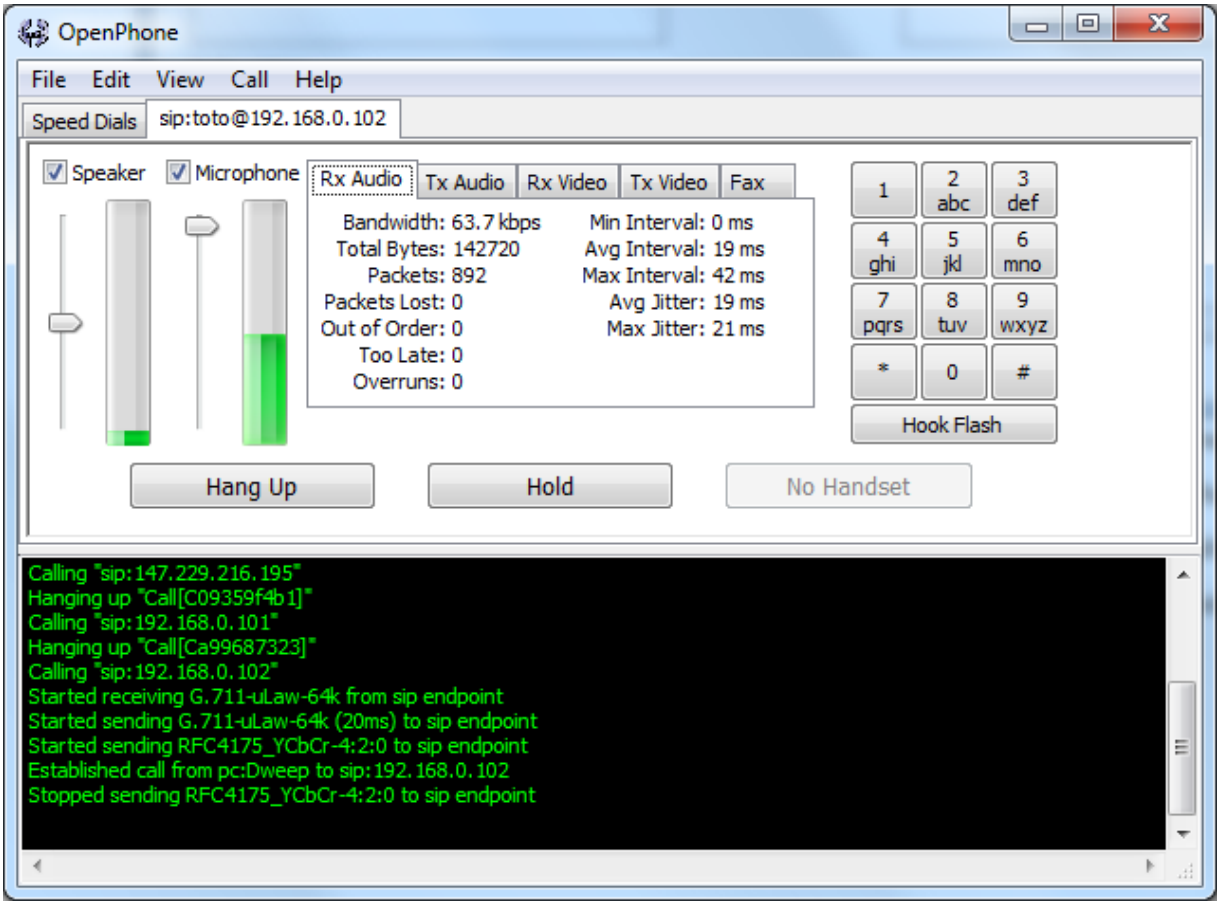
První varianta počítala s použitím vzorového příkladu OpenPhone, který byl výstupem jednoho ze vzorových projektů solution *opal_samples_2008.sln*. Jednalo se o kompletní komunikační zařízení, zahrnující služby pro audio i video volání, registraci i k několika registrar serverům pro konferenční audio hovory a také služby pro Instant

Messaging. Tato aplikace také nabízela velmi rozsáhlé možnosti nastavení a to od různého vzhledu a stylu zobrazení informací přes výběr různých kodeků pro audio či video až po jejich detailní nastavení. Součástí byla také obsluha faxu a IVR (Interactive Voice Response – interaktivní hlasové menu). Samozřejmostí pak bylo nastavení sítě, včetně použití NAT a nastavení statických směrovacích záznamů. Možnosti nastavení jsou zobrazeny na Obr. 9.



Obr. 9 – Možnosti nastavení aplikace OpenPhone – nastavení kodeků

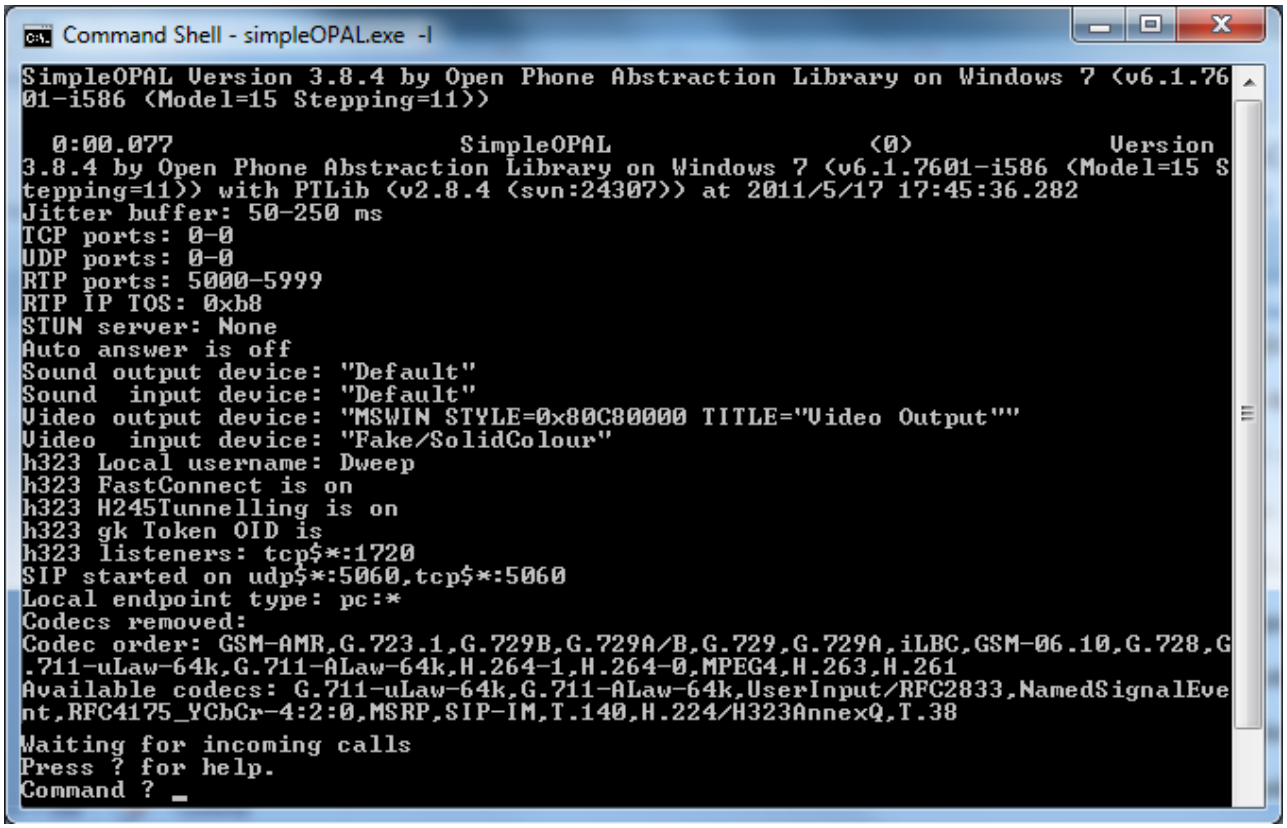
Další výraznou vlastností této aplikace je zobrazování širokého spektra informací o hovoru. Při sestavování spojení jsou v jedné části vypisovány informace o průběhu, během probíhajícího volání je možné sledovat například úroveň hlasitosti přijímaného a odesílaného zvuku, o používaných kodecích a datovém toku (odděleně pro audio a video, příchozí i odchozí). Tyto informace jsou zobrazeny na Obr. 10. Doplňkovou funkcí je ukládání volaných kontaktů do jednoduché formy adresáře.



Obr. 10 – OpenPhone – probíhající hovor

Jak je ale zřejmé z popsaných vlastností, aplikace obsahuje příliš mnoho funkcí, které nejsou cílem této práce. Záměrem bylo vytvořit jednoduchou aplikaci, která bude implementovat pouze protokol SIP a také pouze jeden audio a video kodek. Taktéž zobrazované informace jsou pro zadání nadbytečné. Zvoleným úkolem tedy bylo modifikovat zdrojový kód aplikace a odstranit nepotřebné funkce. Bohužel se zde výrazným způsobem projevila absence komentářů. Kód obsahuje více než 6600 řádků a je takto velice nepřehledný. Používané funkce jsou sice vhodně pojmenovávané, leč jejich závislosti a praktická funkce již popsána není. Navíc je kód propojován s dalšími rozsáhlými hlavičkovými soubory. Proto se jakýkoliv zásah do struktury programu projevil větším, či menším množstvím chyb, které bylo velmi těžké odstranit a ne vždy se to podařilo. Dalším atributem bylo použití zmiňovaných wxWidgets, které sloužily právě pro vykreslování uživatelského rozhranní. Porozumění tomuto nástroji a získání znalostí o jeho použití by vnesly do vývoje výrazné zdržení, proto byla tato cesta vývoje ukončena a práce se věnovala jinému postupu.

Druhá varianta se zabývala opačným přístupem k využití existujícího řešení. Tentokrát bylo záměrem využít velmi jednoduché konzolové aplikace, ke které by bylo nutné vybudovat grafické uživatelské rozhraní. Pro tento postup měl sloužit zmíněný příklad z předpřipravených vzorových projektů, ze stejného solution. Aplikace SimpleOpal se spouštěla z příkazového řádku za pomocí přepínačů a parametrů a s uživatelem pak komunikovala pomocí konzolového okna, viz Obr. 11. Stejně jako OpalPhone podporovala více kodeků pro přenos audia i videa a také protokoly SIP, H.323 i IAX2. Protože zde ale nebylo běžné uživatelské rozhranní a program musel s uživatelem komunikovat textově, stal se zdrojový kód přehlednějším. Také menší množství zahrnutých funkcí a použití přepínačů přispělo ke snazší orientaci a porozumění kódu.



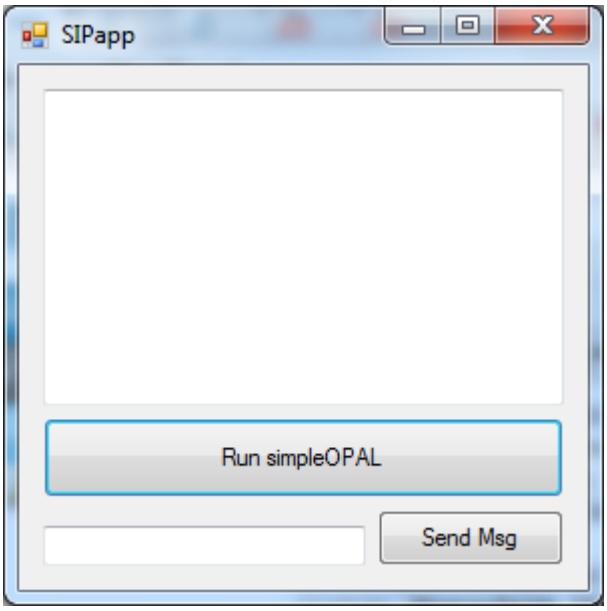
Obr. 11 – simpleOPAL – zkrácený výpis informací po spuštění

4.4 Realizace

Protože bylo nereálné zakomponovat do stávajícího kódu konzolové aplikace prvky grafického rozhranní, byla zvolena cesta vytvoření samostatného projektu pro reprezentaci GUI a propojení těchto dvou programů. Následující postup vývoje lze rozdělit do několika dílčích úkolů:

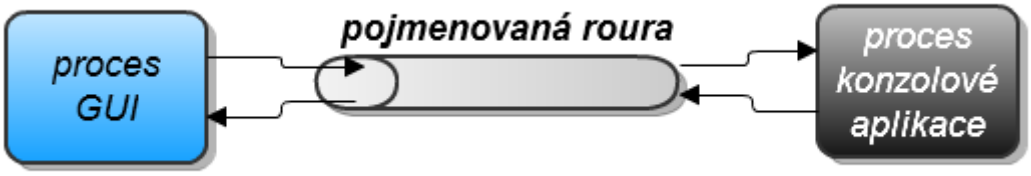
1. návrh jednoduchého GUI
2. vytvoření spojení k manuálně spouštěné konzolové aplikaci
3. automatizace spouštění konzolové aplikace
4. předávání zpráv oběma směry
5. dopracování prvků ovládání klasického SW telefonu
6. reprezentace ovládání GUI do formy příkazů pro simpleOPAL

Prvotní grafické rozhranní bylo určené pouze pro potřeby druhého až čvtrtého úkolu, proto, jak ukazuje Obr. 12, obsahovalo pouze několik prvků. Jedno tlačítko pro spuštění programu simpleOPAL, textové okno s druhým tlačítkem pro zasílání příkazů a druhé textové okno, pro zobrazování zpráv od konzolového programu.



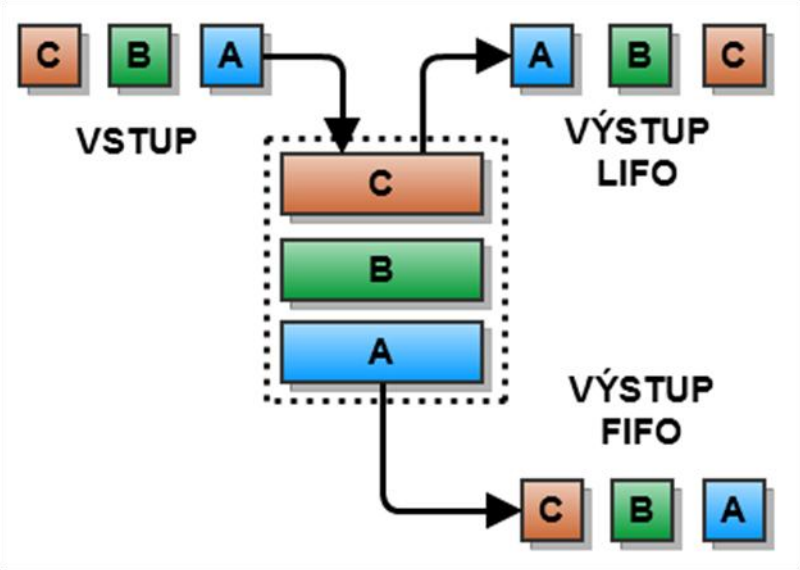
Obr. 12 – Prvotní fáze grafického rozhraní vyvíjené aplikace

Spojení mezi oběma procesy bylo realizováno pomocí pojmenované roury (Named Pipe – NP). NP je specifický objekt, který vystupuje jako propojovací nástroj mezi dvěma procesy, viz Obr. 13. Pracuje jako fronta FIFO (First In, First Out), lze si je tedy představit jako „rouru“, kterou procházejí informace. Co vstoupí na jednom konci jako první, to také jako první vystoupí na konci druhém.



Obr. 13 – Funkce pojmenované roury

Existuje i druhý princip zvaný zásobník LIFO (Last In, First Out), který je možné popsat jako nádobu, do níž se postupně ukládají informace odspodu, jedna vrstva na druhou. Odebírány jsou pak shora, tedy nejprve poslední uložená. Přehledně jsou oba modely zobrazeny na Obr. 14.



Obr. 14 – Princip fronty FIFO a zásobníku LIFO

NP nejsou ve Windows reprezentovány jako speciální soubory (jako například v Unixových systémech), ale jsou to alokovaná místa ve speciálním prostoru určeným právě pro ně. Jsou uchovávaná, dokud práce s nimi není ukončena. Podle [6] mohou být roury polo-duplexní i plně-duplexní, mohou pracovat nejen mezi procesy, ale i mezi počítači s využitím sítě. V této části program pracoval tak, že po vytvoření NP se jeho běh pozastavil, ručně byla spuštěna upravená konzolová aplikace (která se na vytvořenou NP připojila) a následně program pokračoval.

Dalším krokem bylo automatické volání procesu konzolového telefonu z GUI. To bylo realizováno pomocí třídy *Process* ([7]). Ovšem pokud grafický projekt volal další aplikaci, nebylo možné s ním nadále nijak pracovat, dokud volaná aplikace nebyla ukončena. Tento problém se tak řešil pomocí vícevláknového přístupu ([8]). Samo grafické rozhraní bylo spuštěné v prvním vlákně a to také obstarávalo obsluhu pojmenované roury. Pomocí prvního zmíněného tlačítka se nejprve vytvořilo druhé vlákno a v něm se volal druhý proces – konzolová aplikace. Ta tak mohla běžet nezávisle a grafické rozhraní bylo aktivní a umožňovalo zadávání příkazů.

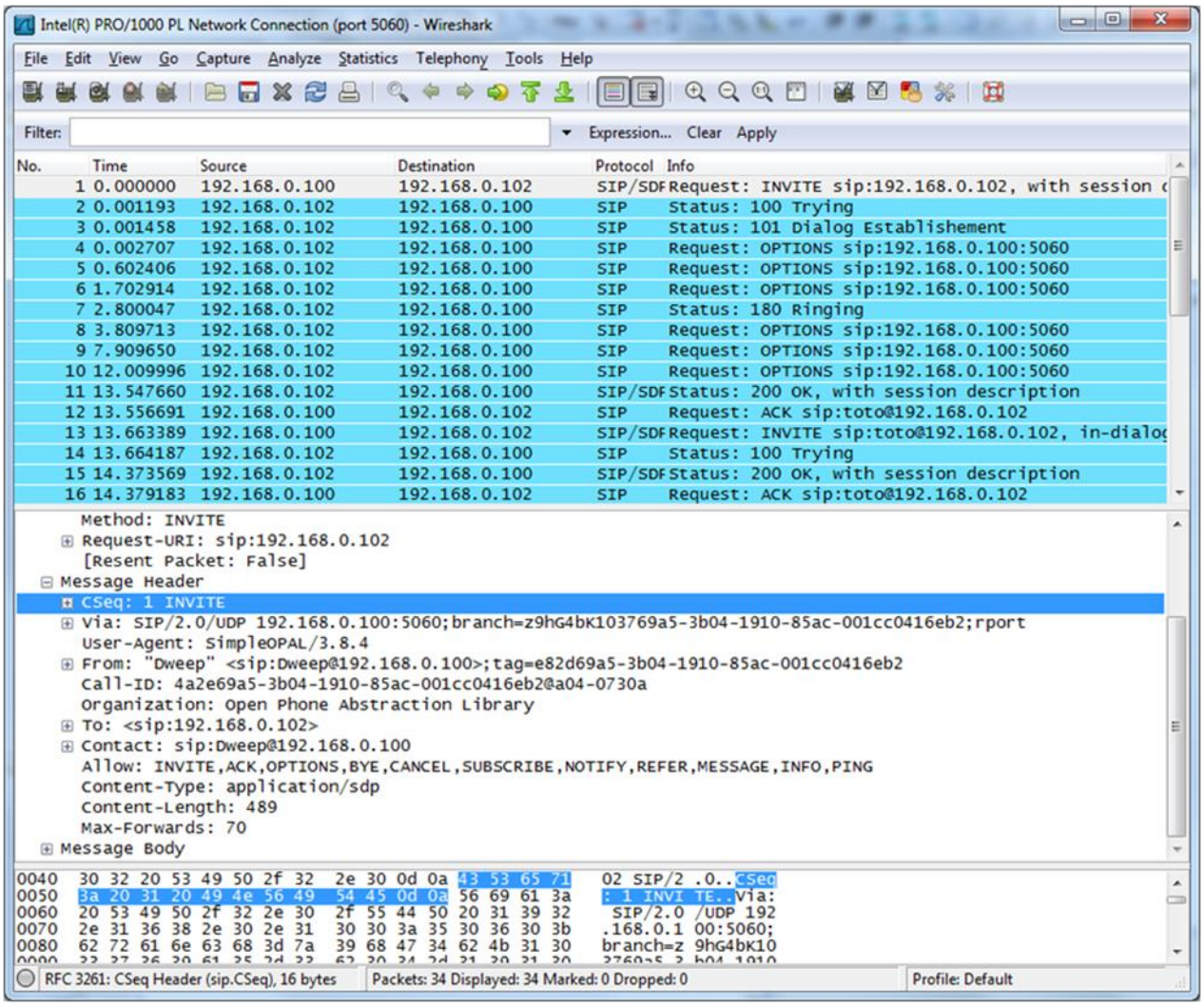
Obsluha roury spočívala nejen v jejím vytvoření a uzavření, ale i v předávání zpráv a také v řízení signalizace. Ta byla použita pro upozornění druhé strany roury, že byla zapsána data (zpráva) a je tedy možné je číst. Předávání zprávy mezi procesy probíhá formou textového řetězce, uloženého v alokované paměti. Protože samotná roura nepracuje v blokujícím režimu, bylo nutné také ohlídat kolizi při zápisu z obou stran roury. Proto se pro tento úkon využívá takzvané kritické sekce, kdy proces nejprve otestuje, zda je roura obsazena z druhé strany. Pokud je, čeká na její uvolnění, pokud není, vstoupí do kritické sekce, a tak zabrání druhé straně v zápisu.

Zde se ale objevil problém s datovými typy a jejich převodem mezi textem z GUI, zprávou předávanou uvnitř roury a speciálním typem, který zpracovával příkazy z konzole. SimpleOPAL totiž využívá vlastní datový typ PString, do kterého bylo třeba zprávy převádět.

V neposlední řadě bylo třeba vyřešit, jakým způsobem předávat výstup z konzolového telefonu do GUI, protože formát projektu neumožňoval standardní čtení z roury. Pro tento směr předávání zpráv byla použita funkce SendMessage s pevně definovanými typy zpráv, které umí třídy grafického rozhranní zpracovávat. Pro přenos právě z konzolového okna se dodefinoval uživatelský typ zprávy, jejíž obsah se pak dále zpracovával v grafickém rozhraní zobrazením v textovém poli.

V závěru práce byla realizována část posledních dvou úkolů. Do grafického rozhraní bylo přidáno jedno tlačítko se symbolem zeleného sluchátka. Jeho funkce spočívala v uskutečnění volání na zadanou IP adresu. Principem bylo vytvoření řetězce začínajícím příkazem „C“, který slouží jako pokyn k uskutečnění volání, následovalo přidání části „sip:“ definující volání s pomocí SIP protokolu, přidání adresy z textového pole a odeslání celého řetězce konzolové aplikaci.

Na Obr. 15 je pomocí programu Wireshark zachycena zpráva INVITE, potvrzující funkčnost grafického ovládání. Dále jsou zachyceny i další zprávy protokolu SIP, popsané v dřívějších kapitolách.



Obr. 15 – Zachycení zpráv protokolu SIP pomocí programu Wireshark – zpráva INVITE

Vzhledem ke komplikacím provázející přípravu a k postupnému upravování směru vývoje, byla téměř vyčerpána doba vymezená pro zadání této práce. Proto byl druhý popsaný postup pozastaven a byla doplněna dokumentace k již odvedené práci. Zdrojový kód i tato práce byla upravena tak, aby mohla posloužit jako základní materiál pro navazující vývoj, v dalším časovém úseku.

4.5 Pokračování ve vývoji

Pro případ pokračování ve vývoji této aplikace, následující text popíše zamýšlený směr pro poslední dva úkoly z nastíněného postupu.

V rámci „dopracování prvků ovládání klasického SW telefonu“ by ke stávajícímu vzhledu byly přidány další prvky:

- tlačítka pro zahájení a ukončení hovoru
- okno pro zobrazování videa v obou směrech (menší pro odchozí obraz, větší pro příchozí)
- tlačítka pro zapnutí a vypnutí vysílání obrazu
- tlačítka pro ovládání zvuku z mikrofonu a reproduktorů
- tlačítko pro otevření okna s nastavením

Je žádoucí, aby se tlačítka pro ovládání obrazu a zvuku realizovala v podobě přepínačů. V rámci zachování přehlednosti a jednoduchosti ovládání nemá být součástí GUI virtuální klávesnice. Není záměrem, aby aplikace vypadala jako běžný telefon, ale aby poskytovala co nejjednodušší rozhraní a zbytečně uživatele nezahrnula nepřeberným množstvím grafických prvků. Nabídka nastavení by obsahovala například výchozí nastavení přepínačů ovládajících zvuk a video, adresu registrar serveru, ke kterému by se po potvrzení dialogu aplikace registrovala.

Posledním prvkem k přidání by byl adresář s kontakty. Jeho realizace by mohla být i v podobě obyčejného seznamu, který by se skrýval stiskem posledního tlačítka.

Druhý úkol se má zabývat reprezentací ovládání pro zpracování konzolovou aplikací. Tento princip byl již zmíněn v předchozí kapitole, pokud tedy uživatel zadá do textového pole např. „192.168.0.101“ do pojmenované roury se zašle řetězec „C sip:192.168.0.101“ a stejně tak i řetězec „marconi@radio.org“ bude pro konzolovou aplikaci doplněn na podobu „C sip:marconi@radio.org“. Obdobným způsobem by tlačítko pro ukončení hovoru zaslalo příslušný příkaz pro přerušení volání a přepínače by ovládaly přenos zvuku a videa.

Další funkce a ovládací prvky jsou na uvážení vývojáře, tyto pokyny jsou pouze pomocné a nejsou nijak závazné. Slouží pouze jako inspirace a pro popis původního záměru vývoje aplikace. Hlavní zásadou by však mělo zůstat zachování co nejvyšší míry jednoduchosti ovládání.

Závěr

SIP patří mezi signalizační protokoly používané pro multimediální přenosy mezi dvěma či více uživateli. Díky své struktuře není multimediální obsah omezen pouze na audio a video hovory, ale je možné přenášet libovolný obsah, například pro herní nebo jiné speciální aplikace. SIP je také možné použít pro Instant Messaging, tedy pro výměnu zpráv v téměř reálném čase. V této diplomové práci byla rozebrána problematika protokolu pro navazování multimediálních spojení. Obecně se práce zabývala strukturou SIP, používanými prvky a prezentovala příklady jejich použití na modelových situacích. Dále pak podrobně popsala dva základní stavební prvky tohoto signalizačního protokolu – žádosti a odpovědi. Žádosti byly rozepsány podle metod, které je definují a bylo uvedeno i jejich použití právě s jednotlivými entitami sítě. Odpovědi byly rozděleny podle tříd a u každé z nich práce popsala jejich obecný význam včetně několika příkladů konkrétních zpráv a použití. Byla nastíněna i část SDP, která jako součást žádosti INVITE popisuje multimediální tok, který chce koncový uživatel zahájit.

V praktické části byl dle zadání vybrán programovací jazyk a také stručně popsány některé knihovny. Z nich byly vybrány knihovny OPAL a PTlib a to z několika důvodů. Knihovny poskytují nástroje jak pro samotný SIP, tak i protokol H.323, zahrnují rovněž nástroje pro implementaci přenosových protokolů pro datové toky audia a videa. Navíc knihovny jsou připraveny pro univerzální použití na mnoha platformách a jejich funkce jsou doprovázeny velmi rozsáhlou dokumentací. Tato volba se ovšem ukázala jako nevhodná. Jediné návody na webu autorů ([5]) jsou pro kompilaci zdrojových kódů pro Windows a Unix. Ty ale nejsou příliš konkrétní, někdy uvádějí zavádějící a nepřesné informace, nejsou relevantní vůči konkrétním verzím operačního systému a odkazují na neaktuální verze doplňkových nástrojů. Dalším nedostatkem volby těchto knihoven se ukázal nedostatek tutoriálů k jejich použití, ukázky jednoduchých zdrojových kódů pro jednotlivé funkce knihoven. Ani zkušenosti ostatních vývojářů nejsou publikovány v obdobné formě. Komplikace, které provázely sestavení knihoven ze zdrojových kódů, zabraly neúměrně dlouhý čas a vývoj samotné aplikace se tomu musel podřídit. Bylo nutné upustit od vytvoření kompletně nové aplikace a namísto toho bylo rozhodnuto využít vzorových příkladů od autorů zmíněných knihoven a jejich úprava dle zadání. Bohužel se do časového termínu nepodařilo aplikaci dovést do

finální verze, ale nedokončená práce byla bohatě dokumentována jak v této práci, tak i ve zdrojovém kódu. Z nastíněných úkolů se podařilo realizovat většinu a z posledních dvou (dopracování grafického rozhraní a zavedení převodu ovládacích prvků do příkazů) alespoň názornou část. Tyto závěrečné úkoly byly časově nejnáročnější, a proto na jejich kompletní realizaci nedošlo. Pokud by bylo v práci pokračováno, byl nastíněn směr, jakým se u zbylých úkolů ubírat.

Hlavním přínosem této diplomové práce je detailně vysvětlená struktura protokolu SIP a jeho užití, krátce popsané některé knihovny pro implementaci a zejména prezentovaný návod k bezproblémovému zkompilování dvou vybraných knihoven. Dále bylo prezentováno několik možných přístupů k vývoji aplikace dle zadání. Podle časových podmínek pak byl vybrán jeden z nich a ten postupně realizován. Protože však nemohl být dokončen, byl rozpracovaný projekt dokumentován a byl nastíněn záměr případného budoucího vývoje. Je tedy možné v započaté práci pokračovat, nebo při vývoji kompletně nové aplikace se lze vyhnout zmiňovaným potížím.

Seznam použité literatury

[1] RFC 3261. *SIP: Session Initiation Protocol*. East Hanover: Internet Engineering Task Force, 2002. 269 s.

[2] *JOHNSTON, Alan B.* SIP: Understanding the Session Initiation Protocol*. Second Edition. Boston: Artech House, 2004. 283 s. ISBN 1–58053–655–7.*

[3] FIRESTONE, Scott; RAMALINGAM, Thiya; FRY, Steve. *Voice and Video conferencing Fundamentals*. 1st ed. Indianapolis: Cisco Press, 2007. 376 s. ISBN 978–1–58705–268–2.

[4] REGUEYRA, P. SIP klient pro Windows Mobile. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 44 s.

[5] *OpalVoipWiki* [online]. October 31, 2007, November 16, 2010 [cit. 2011-04-15]. Building PTLib on Windows. Dostupné z WWW: <http://www.opalvoip.org/pmwiki/pmwiki.php?n=Main.BuildingPTLib>.

[6] Microsoft. *Named Pipes (Windows)* [online]. 1/27/2011 [cit. 2011-05-18]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/aa365590.aspx>.

[7] Microsoft. *Process Class (System.Diagnostics)* [online]. 2011 [cit. 2011-05-18]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/system.diagnostics.process.aspx>.

[8] Microsoft. *Thread Class (System.Threding)* [online]. 2011 [cit. 2011-05-18]. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/system.threading.thread.aspx>.

Seznam zkratek

ACK	ACKnowledgment	Potvrzení
GUI	Graphic User Interface	Grafické uživatelské rozhraní
IAX2	InterAsterisk eXchange	Komunikace mezi ústřednami Asterisk
IETF	Internet Engineering Task Force	Komise techniky Internetu
IVR	Interactive Voice Response	Interaktivní Hlasové menu
NAT	Network Address Translation	Překlad síťových adres
NP	Named Pipe	Pojmenovaná roura
SIP	Session Initiation Protocol	Protokol navázání spojení
SDP	Session Description Protocol	Protokol popisu spojení
RFC	Request For Comments	Popisný dokument protokolu
RTP	Real–time Transport Protocol	Protokol přenosu v reálném čase
RTCP	RTP Control Protocol	Protokol řízení RTP
SCTP	Stream Control Transmission Protocol	Protokol kontroly proudového vysálání
TCP	Transmission Control Protocol	Protokol kontroly vysílání
UAC	User Agent Client	Klientská část uživatelského agenta
UAS	User Agent Server	Serverová část uživatelského agenta
UDP	User Datagram Protocol	Datagramová služba
VoIP	Voice over Internet Protocol	Volání přes internetový protokol

Obsah CD

- /knihovny – obsahuje 4 zkompilované knihovny
- /SIPapp – obsahuje vytvořenou aplikaci SIPapp, podpůrnou aplikaci simpleOPAL a textový soubor ReadMe.txt, který obsahuje návod k použití aplikace
- /zdroje – obsahuje zdrojové soubory knihoven a nástroje pro kompilaci, NEobsahuje Microsoft Visual Studio

ReadMe.txt

Hlavní aplikace se spouští souborem SIPapp.exe.

Ovládání:

- Tlačítko „Run simpleOPAL“ spustí pomocnou aplikaci (pro kontrolu se zobrazeným konzolovým oknem) – aplikace pak čeká na příkazy nebo na příchozí volání.
- Tlačítko „Send Msg“ odesílá text v podobě, v jaké je v sousedním textovém poli – umožňuje zadávání přímých příkazů.
- Tlačítko se zeleným symbolem sluchátka provede volání na IP adresu zadanou v klasickém formátu „AAA.BBB.CCC.DDD“. Není třeba přidávat žádná klíčová slova ani příkaz pro volání, to obstará program sám.